IN THE SPECIFICATION:

Please amend paragraph starting at line 19, page 2, as follows:

The present invention is also directed to a method for serializing data sent through a high performance interface of a network switch, where the method includes the steps of receiving parallel data to be sent over the high performance interface and multiplexing the parallel data. Then a portion of the parallel data is stored in a first register clocked on a positive edge of a clock signal and another portion is stored in a second register clocked on a negative edge of the clock signal. The portions are then input into first level glitchless multiplexors and the portions are multiplexed based on a ~~mutliplexor~~ multiplexor selection signal input into those first level glitchless multiplexors. The outputs of said first level glitchless multiplexors are ~~ouput~~ output to a second level glitchless multiplexor and the data is multiplexed by selecting alternating inputs based on the ~~mutliplexor~~ multiplexor selection signal input into the second level glitchless multiplexor. Each of the first level glitchless multiplexors produces a function hazard when more than one input to the first level glitchless ~~mutliplexor~~ multiplexor changes simultaneously. The multiplexing of the data by the second level glitchless multiplexor is timed such that the second level glitchless multiplexor only selects input from one of the first level multiplexors that is not producing a function hazard.

Please amend paragraph starting at line 28, page 43, as follows:

The present invention provides a clear advantage over single address table lookup schemes, as the great majority of the concurrent searches are conducted in parallel. Therefore, in the case where both searches require only comparisons, performance doubles, irrespective of address insertions and deletions, as these operations are of a lower priority and have no ~~affect~~ effect on performance. As a specific example, performance for the parallel operation is calculated by multiplying the number of cycles per search by the number of clock cycles per search cycle, and then adding the clock overhead. This calculation is represented by the following equation:

Performance=(#cycles parallel)·(2clocks/search cycle) + overhead (1)

Therefore, the performance for an 8k table using the present invention is represented by the following:

(13)·(2)+4 = 30 clock cycles for 2 packets or

15 clock cycles for a single data packet (2)

Performance for a 16k table using the present invention is represented by the following:

(14)·(2)+4 = 32 clock cycles for 2 packets or

16 clock cycles for a single packet (3)

If only a single data packet requires an address lookup, the following represents the search time to accomplish the lookup with the present invention:

(13)·(2)+4 = 30 clock cycles per packet (4)

(14)·(2)+4 = 32 clock cycles per packet (5)

Please amend paragraph starting at line 10, page 55, as follows:

In executing actions from the rules table entries and no match actions from the filter, specific rules are followed in order to insure proper filtering and action execution. The relevant rules to execute actions from rules table entries and no match actions from filters are as follows.

- When a binary search is done in the rules table, additional comparison is done using {filter select + egress module id + ingress port + egress port} fields to determine a partial match
- A full match occurs when the filter select + egress module id + ingress port + egress port + packet format + filter value matches an entry in the rules table. Therefore, if there is a full match, then the associated actions from the matched rules table entry are applied.
- If there is no full match and no partial match, then no action is taken.
- If there is no full match, but there is a partial match, then the actions from the no match actions field are applied. This no match action is derived from the filter mask field.
- If there is a partial match with a filter, actions associated with the filter mask are taken. If there is a full match with a higher filter value, then the actions associated with the rule entry are taken. If a particular action bit is set by the no match action field and the full match on another filter mask

does not set the same action bit, then the action is taken, as the partial match and full match are on different filters.

A3 Cont

• If there is a partial match and a full match, the counters are updated only for the full ~~march~~ match according to the rules table. If there is only a partial match, then the counters are updated according to action in the filter mask. If all of the filters have a full match in the rules table and the action is to increment the same counter, then the counter is incremented only once. If all of the filters have a partial match and the action is to increment the same counter, then the counter is incremented only once.

Please amend paragraph starting at line 22, page 58, as follows:

A4 In using multi-field classifiers, SOC 10 uses FFP mechanism 141 to implement the multi-field (MF) classifications, which are accomplished at the network boundaries. MF classifier capability is implemented using the FFP 141 engine, wherein the filter mask and the corresponding rules table are ~~programed~~ programmed as per the corresponding Differentiated services related policies to assign a new code point or the change the codepoint of the packet. The same rules entry can be used to change 802.1p priority of the packet, depending on the particular policy.

Please amend paragraph starting at line 12, page 62, as follows:

Figure 47 shows a detailed flowchart of the logic contained within step 42-4 of Figure 42. At step 47-1 the logic gets the PortBitmap and conducts a logical "and" operation with this value and the forwarding port register, while also "anding" this value with the active port register corresponding to the COS queue selected, after going through the COS mapping using the COS mapping using the COS Select Register. This value is also "anded" with the HOL Register value, which corresponds to Active Port Register 8, to get the PortBitmap at this step. The logic also looks at the M bits of the port based VLAN at this step. Upon completion of the actions of step 47-1, the logic continues to step 47-2, where the logic checks to see if the ingress port is mirrored, that is if the M bit is 0, or if the stack link and the M bit is set. If so, then the packet is sent to the according mirrored port at step 47-3, while if not, then the logic continues to step 47-4 without taking any mirror port action. At step 47-4 the logic checks to ~~se~~ see if the mirroring is based upon filter logic. If so, then the packet is sent to the appropriate mirrored port at step 47-5, while if not, then the logic continues to step 47-6 without taking any mirrored port action. At step 47-6 the logic checks to see if the egress port is mirrored by looking at the egress mirroring register. If the egress port is mirrored, then the packet is sent to the mirrored port before continuing to step 47-8. If the packet is not mirrored, then the logic simply continues directly to step 47-8 without taking any mirror port action. Step 47-8 continues with the mirror port logic, and therefore, will not be discussed in detail. Nonetheless, at this stage of the logic the DSCP has been accordingly modified such that the packet flow can be appropriately shaped and/or metered upon egress.

Please amend paragraph starting at line 8, page 79, as follows:

Figure 25 illustrates a flowchart for how ingress 14 of an SOC ~~20~~ 10 would handle an IP multicast packet coming in to a port thereupon. In step 25-1, the packet is examined to determine whether or not it is an IP multicast packet without any option fields. If there are option fields, the packet is sent to CPU 52 for further handling. In step 25-2, the IP checksum is validated. In step 25-3, the destination IP address is examined to see if it is a class D address. A class D address is one where the first three most significant bits are all set to 1. If the destination IP address is not a class D address, then the packet is dropped. If so, the IP multicast table is searched at step 25-4 with the key as the source IP address + the destination IP address. If the entry is not found, then the packet is sent to CPU 52. If a match is found at step 25-5, the TTL (time-to-live) is checked against the TTL threshold value in the IP multicast entry at step 25-6. If the TTL value is less than the threshold value, the packet is dropped. If the TTL value is not below the threshold, then the source port is compared to the source port in the entry at step 25-7. If there is not a match, then the packet is dropped. If there is a match, the packet is appropriately sent over C channel 81 of CPS channel 80 at step 25-8, with appropriate P channel messages locked therewith. The packet is sent with the L2 port bitmap and L2 untagged bitmap obtained as a result of the L2 search, and the L3 port bitmap as a result of the IP multicast search. Additionally, the IP multicast bit is set in the P channel message, indicating that the packet is an IP multicast packet and that the egress, upon receipt of the packet, must modify the IP header appropriately. From CPS

channel 80, therefore, the packet is sent to the appropriate buffer pool until it is obtained by the appropriate egress port.

Please amend paragraph starting at line 23, page 94, as follows:

Referring to Figure 29, address resolution for a packet coming in to IPIC 90 from high performance interface ~~271~~ 261 is as follows: